# Drive Interface Design Rev 1.0

## **Goal:**

Part 1 is a group activity where you will discuss and define an interface for your piRover motion. First you will discuss the action the user will take using the keyboard to enter drive commands. You will need to specify keyboard inputs required for each motion listed in Table 1. Additionally, you will need to consider keystrokes required to control the speed of the motion.

In part 2, you will create the Python program that will capture the inputs specified in part 1 and simply print a message indicating the type of motion and speed that is being requested.

In part 3, you will create a piRoverDrive.py file that contains the definitions for the motion functions that include speed. You will not implement the motion function at this point, just defined the function and use a print statement as an output to the terminal window verifying with the function has been called.

## **Prerequisites:**

This assessment requires content and code solutions from the following:

* User Move
* PWM Introduction

## Performance Outcomes:

1. Design a keyboard interface for piRover motion that includes speed.
2. Create user input and selection code that implements interface design.
3. Create piRover Drive function “stubs” that define functions but do not implement piRover motion.

## Resources:

1. See prerequisite lessons

## Materials:

1. piRover

## Set Up

1. Prepare your workspace for this project.
   1. Connect to your piRover using VNC. Access your piRover folder and launch VS Code.
   2. Create a **10.KeyboardDriveDesign** directory.
   3. Create a new **keyboard\_drive.py** file in the 10.KeyboardDriveDesign directory.
   4. Create a new **piRover\_drive\_fake.py** file in the 10.KeyboardDriveDesign directory.

## Part 1 – Keyboard Interface Design

1. Your task is to work with your group to define a keyboard interface for piRover motion.
2. Review the required motions in Table 1 and consider that speed must be controlled during each motion.
3. Discuss with your group the use of the input() function and how keystrokes along with the Enter Key can be used to create a drive interface. Enter the keyboard input in Table 2 that is required for each motion. Be sure to incorporate speed control in your interface.

Table 1

|  |
| --- |
| Drive Interface Requirements |
| Forward |
| Backward |
| Left Turn |
| Left Rotate |
| Left Pivot Back |
| Right Turn |
| Right Rotate |
| Right Pivot Back |
| Stop |
| Set Speed |
| Accelerate |
| Decelerate |

1. Enter your moves in column 1 of table 2. These can be a copy of table 1 requirement or something more complex. You must include all motion and speed requirements in table 1.
2. Enter the key input required in column 2 for the movement identified in column 1. Discuss specifics with your group. How will keys and either single or multiple input() statements be used to create specified motions. An example of Forward is provided, but how will speed be specified with this most basic example? Delete the example and enter your group’s interface design.

Table 2

|  |  |  |
| --- | --- | --- |
| **Movement** | **Keyboard Input** | **Function Call(s)** |
| (sample only)  Forward | W | Forward() |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. Now create and list the required drive methods in column 3. Can you specify a single drive function and use parameters to define the required detail of the move? Work to create good naming for both move function and associated parameters.
2. Be prepared to share your design with the class.

## Part 2 – Keyboard Drive

1. Create the **keyboard\_drive.py** code that implements the user interface defined in Table 2. This is the “main” solution file that does the following
   1. Welcomes the user to the keyboard move solution
   2. Provides directions to the user including a list of keyboard commands
   3. Captures the command
   4. Use a selection structure to print an appropriate movement action.
   5. Includes a method for the user to exit the solution
   6. No move action is required. This comes later.

## Part 3 – Drive Fake

1. Create the **piRover\_drive\_fake.py** code that does the following
   1. Defines the motion functions required for table 2.
   2. Use parameters as required.
   3. Create good naming of functions and parameters.
   4. Include an init() function.
   5. Use print statements to display the movement or initialization action.
   6. **No move action (GPIO coding) is required**. This comes later.
   7. Add sample code at the bottom of this file that calls init() and executes a variety of move and speed operations. This will be used later to test your drive solution.

## Submission:

1. Submit this document with Table 2 along with keyboard\_drive.py and piRover\_drive\_fake.py to the Moodle site using the links provided. This is an in-class, group activity.